Awareness on the Roads Through Video Analysis

Undergraduate Honors Research Thesis

Presented in Partial Fulfillment of the Requirements for the Degree B.S. Computer Science and Engineering with Honors Research Distinction in the College of Engineering at The Ohio State University

By

Erik Petersen Schilling

B.S. Computer Science and Engineering

The Ohio State University

2014

Committee:

Prasun Sinha, Advisor

Chunyi Peng

Abstract

Distracted driving is a common factor in many accidents in the United States. Car manufacturers are trying to implement technologies to increase a driver's awareness of the road. While systems such as lane departure warnings, blind spot monitors, and adaptive cruise control are effective, there are few third party technologies and essentially none that are open source. The system described in this thesis uses a robust, cheap, and accurate vehicle detection system that is extendable, allowing for other parties to contribute to the system.

The system detects key vehicle characteristics in a tiered process. Each tier prunes the set of hypothesized vehicle regions by requiring each region pass more refined characteristic criteria. Lane detection is also leveraged to reduce vehicle detection time. Testing shows that the system has a precision of 82% and a recall of 76%.

Acknowledgments

I would like to thank Prasun Sinha, my research advisor, for all the time and experience he's contributed to making my research successful.

Vita

2010-Present . . . . . Ohio State University Honors

2010-Present . . . . . Maximus Scholarship

2012-Present . . . . . Kroeger Scholarship

June 2012-August 2012 . . . . Cisco Choice University Engineer Intern, Cisco Systems

Fields of Study

Major Field: Computer Science and Engineering

Table of Contents

List of Figures

1 Introduction

Every 30 seconds some form of driving accident occurs in the USA [1]. 20% of these crashes in which someone was injured involved distracted driving [2]. Some car companies are trying to reduce this number by providing customers with ways to supplement their driving with computer based surveillance of the road. The technology ranges from blind spot detection to adaptive cruise control to lane departure warnings [3]. All of these devices have had success in improving driving conditions, however, there is still desire to provide a system that incorporates different sets of data to provide a more complete view of the road.

One such system, utilized heavily by autonomous vehicles like Google's self-driving cars [4], is a laser/radar (LIDAR) array. This array provides the vehicle with a 360 degree view of the surrounding area. While this system provides high fidelity, allowing for accurate object distance and size calculations, the price-tag of $70,000 is much too high for the average consumer.

All of these devices have been able to increase driver safety; however, there are some issues that are presented when third parties wish to leverage this technology for their own means. The main problem stems from the fact that these solutions are based on expensive, job specific equipment and paired with proprietary software that prevents other companies, institutions, or individuals from improving on or learning from the device.

We have developed an open source vehicle detection and tracking sys-

tem that requires only the software and a dashboard mounted smartphone camera. The system is robust enough to allow for a wide array of cameras and camera placement as well as various road and lighting conditions. The system can perform under varying cloud coverage, however, it cannot function during night hours and must be used during the day time. This method allows for outside entities to utilize the developed software without the need for a use-specific hardware setup or be restricted in testing by environmental conditions.

The software was written mainly using the opensource libraries from OpenCV with a C++ interface [5]. OpenCV was developed by Intel and is now supported by Willow Garage and Itseez. The library functions were designed for computational efficiency and focus on computer vision and machine learning. OpenCV is geared towards real-time applications and thus was a good fit for the requirements of this research.

2 Literature Review

2.1 Detection Methods

Vehicle detection has been a popular topic for improving driver safety and overall driving experience. Because of this, many different systems have been developed by multiple groups with published results. This section will describe the various methods researched and the following section will analyze the methods.

The method used in [6] is different for recognizing passing and distant vehicles. Passing vehicles cause a sudden change in illumination as well as cover a significant portion of the frame. By tracking the change in illumination from past frames, the software can approximate where the vehicle is in the current frame. This method does not work for distant cars as there is no significant change in illumination or motion between frames. Instead, a two step process is employed for detection.

In the first step, a coarse search, the entire frame is scanned for prominent vertical and horizontal edges. If a region is found with sufficiently large edges it is determined to possibly contain a vehicle and is moved to the refined search. In this step, each region is placed on a vertical gradient and then on a horizontal gradient. This process reveals prominent vertical and horizontal edges respectively. If a pair of sufficiently prominent horizontal edges and a pair of vertical edges are found, then it creates a bounding box for the potential vehicle.

The next method, described in [7] also uses the horizontal and vertical features of vehicles for recognition.

In [8] and [9], libraries of vehicle templates are used to compare with hypothesized regions in order to identify the vehicles.

The method described in [10] uses inverse perspective mapping (IPM) to identify vehicles. IPM involves transforming a perspective view image into a birds-eye view image. This method removes the effects of perspective distortion which causes objects to move in a non-linear fashion when captured from a camera. In order to detect vehicles, a road removal process is performed before the image is mapped. This process involves calculating the average red, green, and blue values of the area immediately in front of the vehicle. This area was chosen because it is most likely empty road. After the three values are calculated, the algorithm loops through every pixel in the frame and compares it to the road values. If the pixel is within a certain threshold it is considered road and replaced with a black pixel, otherwise it is replaced with a white pixel. Three images are created during this process, one for red, green, and blue. Next, the images are combined into a final image that is then mapped to a birds-eye view. The resultant image provides a useful visualization of the region in front of the vehicle and allows for easy vehicle detection as well as distance estimation.

There are more methods that are described in the review from Sun et. al. [11]. Among which is a method that uses two cameras to create a stereo vision view of the region in front of the car. This method creates a disparity

map that allows for a 3-D map of the viewed scene to be created. This allows for easy differentiation between vehicles and the background. However, there was no pointer to an implementation of this idea for detecting vehicles. One source in [12] uses the method to detect road debris while another, in [13], discounts the method entirely as being too computationally inefficient.

2.2 Analysis of Methods

The following section will discuss the advantages and disadvantages of each method described in the previous section.

The first method described was from [6] in which a multi-step process was used to detect vehicles at a distance. This method allows for short process times because the time consuming "coarse step" is run once every ten frames while the more in depth "refined step" is run on small regions of interest. However, a disadvantage of this method is that most vertical edges are not picked up in the refined step. This is because cars do not actually have vertical edges but rather sloped slides that will not appear in a vertical gradient. Also, in the coarse step, many false positives may be selected such as billboards, buildings, or windows due to the lack of limiting characteristics to select regions of interest. This flood of false positives will unnecessarily slow down the refined step and reduce overall processing speed.

The method described in [7] can also return false positives because of this reason.

In both [8] and [9] the required libraries were too large to be generated

for the scope of this research. It is possible to use a more general template to reduce generation time. A template as simple as locating the license plate and back windshield would be reasonable to create for this project, however, the error rate associated with this template would be incredibly high as any object in the frame that is remotely rectangular would be returned as a possible vehicle.

The inverse perspective mapping method described in [10] provides an effective process for detecting vehicles as well as easily calculating the distance between the user's car and the next vehicle in their lane. However, the disadvantages of this method are two fold. One, is that the hardware system must be calibrated specifically for each vehicle. This can be done manually by displaying a 1 meter white stripe in the line of sight of the camera, along a flat surface, to allow for the measurement of the number of pixels equivalent to 1 meter. The calibration could be done automatically by placing a chessboard pattern of known dimensions in front of the camera to determine relative dimensions. Lastly, the calibration could be done on-the-fly by using road markings as measurement indicators. The first two methods require very specific hardware configurations that do not lend themselves to easy recreation by other researchers or individuals. The lane marker approach is more robust when it comes to hardware setup, however, it does rely on the assumption that all lane markers are of the same size. Also, the inverse perspective mapping itself requires the user to know the height, angle, and focal length, among other characteristics, of the camera in order to correctly map.

The second major disadvantage is the fact that this method only detects the bottom of vehicles and has difficulty finding the tops due to distortion near the top of the mapped image. This method is useful for distance estimation but is less successful when it comes to specific vehicle detection.

2.3 Existing Applications

There are some groups that have apps or standalone devices that can track vehicles. iOnRoad is an app with a free and paid version [14]. The app has many useful features including user speed, lane departure warning, and distance from next vehicle. However, one important shortcoming of the app is that it only tracks a vehicle if it is in the same lane as the user. A company with a standalone device is MobileEye [15]. This device is able to perform multi-vehicle detection, lane detection, and distance detection. However, the prices range from $750-$850, impractical for the average consumer.

3 System Design Overview

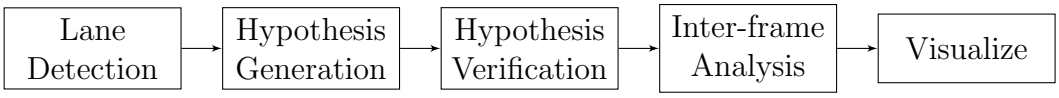

Figure 1: Overview of the system implementation

The system attempts to provide a robust and fast method for detecting and tracking vehicles. It uses mounted cameras to detect vehicles in its field of view as well as the lane markers on the road. Currently, the system assumes only one camera per car is used but it can be updated to include more cameras for better identification of vehicles.

The design of the system is as follows. Lane markers are the dashed or solid lines used to differentiate between lanes on the road. These markers are first detected to create a more refined search area for vehicles. This is accomplished mainly through edge detection. Next, the vehicles are detected in each frame using a tiered system of known vehicle characteristics. Each tier is more refined than the previous one which allows for reduced processing time for each region detected. Last, each vehicle is tracked from frame to frame in order to continue detecting missed vehicles as well as reduce the number of false positives that are detected.

3.1 Lane Detection

Lane detection is an effective way to remove many of the false positives that would have normally been detected when searching for vehicles. The algo-

rithm is able to determine if the lane detected is a solid line or a dashed line. This information allows the vehicle detection algorithm to ignore anything that is outside of solid lines (meaning it is off the road) but still identify vehicles in neighboring lanes. Lane detection searches for all edges in an area in front of the user's vehicle. The list of lines returned is then pruned of all lines that could not indicate a lane marker due to length, location, or angle. Multiple lines from the remaining list can identify the same lane marker so all lines are sorted and combined based on proximity to each other. The resulting lines are sufficient to identify each lane marker in the frame as well as contain information that informs the vehicle detection algorithm as to which type each lane is identified as.

3.2 Vehicle Detection Algorithm

Vehicle detection relies on a tiered system of key characteristics to identify vehicles in each frame. The first identifier used is contrast between the bottom of the vehicle and the road. During preliminary work it was discovered that the region directly beneath the car was significantly darker than the rest of the road even during cloudy conditions. These dark areas provide a region in which more computationally intensive algorithms can be run more efficiently. The region is determined based on of the size of the dark area identified. The next identifier determines if the dark areas are symmetric enough to be considered vehicles. This is a useful identifier because rear views of cars are symmetric while tree lines (the most common false pos-

itive identified in the previous tier) are not. Next, each region of interest is scanned to determine the edges of objects in the region. From this scan, we try to identify the top of the vehicle. This is accomplished by searching for horizontal lines in the top half of the region of interest. The length of each horizontal line is scaled based on of its distance from the middle of the image, which is the expected location of the top of the car. This technique prevents the algorithm from incorrectly identifying the horizon or a bridge as the top of a vehicle simply because it is a long straight line in the region. If no sufficiently long line is found it is determined that the region does not contain a vehicle and is removed. Finally, any remaining regions are returned as containing one vehicle.

3.3 Vehicle Tracking

Vehicle tracking provides an added level of confidence that the regions detected are, in fact, vehicles. Tracking is done by comparing the current frame with the previous frame. After the vehicles have been detected in the current frame the results are compared to the information from the previous frame. If a region is detected in the current frame but not the previous frame it could indicate that the region is a false positive so it is not displayed in the output but still stored for the next frame. If a frame was detected in the previous frame but not the current frame it could be simply because the current frame missed it so it is displayed. However, there is a score associated with each region in order to prevent a continuous accumulation of regions even

after the vehicle has left the frame, so in the instance described, the score of the region would be decremented to indicate the lack of confidence due to the disparity between frames. Finally, if both frames identify the same region the score of the region is incremented to indicate the confidence that the region does contain a vehicle.

4 System Implementation

4.1 Lane Detection

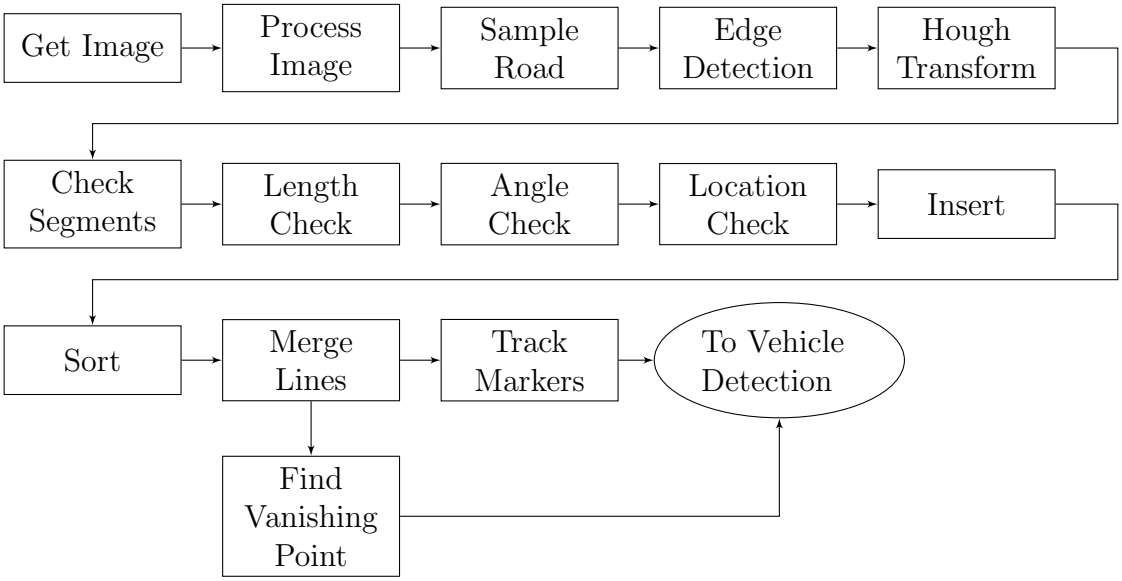

Figure 2: Lane Detection

4.1.1 Process Image

Lane Detection begins by processing the current frame. To do this, a region of interest is found in the image that is directly in front of the vehicle and contains a view of the road up to about twenty feet in front of the vehicle. The reason that only this portion of the image is viewed is because lane markers cannot be detected much further out as they begin to blend in with the surrounding road due to video quality so any edges detected at that range are just noise. Once the region has been selected, a Canny edge

detector is used from the OpenCV library. This function detects the edges in an image by comparing neighbor pixel values and any pixels that show a sharp enough contrast with their neighbors are considered edge pixels and are marked. From this image, the Hough Line Transform function is run to find the edges that are long enough to be considered straight lines. The Hough Line Transform function works by detecting the polar coordinates of every line that intersects every edge pixel. These curves are then compared across pixels and any that intersect provide the radius and angle of the polar coordinate line that intersects both pixels. There is a threshold value that can be placed to require a minimum number of intersecting curves to be present to consider that curve to contain a straight line.

4.1.2 Check Line Segments

Now, there is a list of potential lane markers that must be checked to remove any false positives. The three requirements that each line segment must pass are length, angle, and location. First, the line must be long enough. Results from the Hough Line Transform may have been found near a dense cluster of pixels in which a high number of curves were found even though the separation of pixels was small. Such segments are removed due to not meeting a minimum length requirement. The second requirement checks the angle of the line segment. All lane markers are oriented at the same point in an image, this is due to the characteristic known as the vanishing point for perspective images. Therefore, any hypothesized lane marker must

point “near enough” to the vanishing point to be considered part of the lane. This turns out to be the most critical characteristic of the lane markers and ends up removing most of the line segments from the Hough Line Transform results. Calculation of the vanishing point is discussed later. Lastly, the location of each line segment is checked; more specifically, we check that the line segment is in fact the edge of a lane marker and not a crack in the road, a shadow, or the edge between the edge of the road and the grass. This check is done by comparing the left and right side of each line segment to make sure that one side contains pixels that are “close enough” to lane marker colors and the other side contains pixels that are “close enough” to road colors. If both sides do not meet this requirement the segment is rejected. This check is the most processor intensive of the three which is why it occurs last since most false positives have been removed by this point.

4.1.3 Sort Segments

The results of the three checks are considered the lane markers of the road. However, due to the nature of edge detection and line transforms, multiple line segments can be found for the same lane marker. Because of this the segments must be combined into one lane marker. To do this, the lines are sorted and combined based on distance from each other. This is done by inserting each line segment into a binary tree based sorting algorithm as it passes the previous three requirements.

4.1.4 Combine Segments

The line segments are sorted based on the x-coordinate of the line at the bottom of the image. This metric was chosen because the lines will later be combined based on this value and the bottom of the image is used for calculations because this is the position where the highest variance occurs, providing better fidelity when comparing the line segments. Once all segments have been inserted, they are combined. This process is done by starting with the left-most line segment and averaging the slope and bottom x-intercept of each line that is within a certain range of the average line that is being generated. Once all nearby lines have been combined, the resultant line is added to a list of lane markers and the process is repeated until there are no lines left. The resulting lane markers from this process are then used in a tracking algorithm that is used to smooth out lane markers between frames.

4.1.5 Calculate Vanishing Point

Once the lane markers have been determined the vanishing point for that frame can be calculated. This is done by finding the intersections of all lines since they should intersect at the vanishing point. The resulting set of points is then averaged to find the vanishing point for that frame. Previously, it was discussed that line segments are removed if they are not close enough to the vanishing point. The point used in this calculation is the average vanishing point for the video. This value is calculated by weighting the current vanishing point and combining it with the previous average.

4.1.6 Marker Tracking

Finally, the lane markers are placed in bins which aid in providing a more uniform and smooth visualization of the lanes as the car moves. This algorithm looks at each lane marker and compares it to each of the bins, if the x-intercept with the bottom of the image for both the bin and lane marker are close enough to a specified threshold or the bin is empty then the lane marker is placed in the bin. Although, if the x-intercepts are too close, then the lane marker is not inserted. These two features allow for the lane positions to update as the car moves or changes lanes but reduces small movements that are created due to different calculations in determining lane markers. Currently, there is a hard count of 4 bins that lane markers can be placed into. This is because the range of lane detection does not have the range to extend past one lane to either side. However, if that were to change, adding bins as needed would be a possible feature to add.

4.2 Vehicle Detection

4.2.1 Process Image

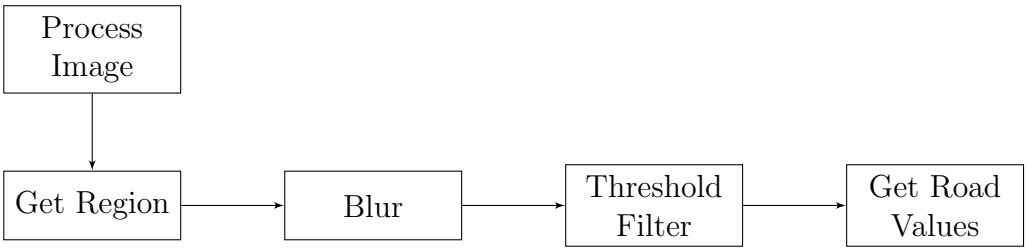

Figure 3: Vehicle Image Processing

Similar to lane detection, vehicle detection begins by processing the current frame. This processing is different in that the region of interest of the image is bound on top by the vanishing point from lane detection and on the bottom by a constant value that makes sure the hood of the car is not considered in further processing. This image is then blurred in order to reduce noise. From this point, the image is run through a threshold filter. This algorithm scans the road in front of the car to calculate road illumination. This provides the algorithm with a sense for how well the entire image is illuminated. This value is important because it will be used to detect the shadowy area that exists beneath all vehicles given even a minimal amount of daylight. The illumination value is calculated and placed in a predetermined formula to provide the threshold value for the image. Next, a threshold filter is passed over the image and any pixels that are below the threshold are turned black and any pixels above are turned white. This resulting image is then used in the next step.

4.2.2 Contour Checks

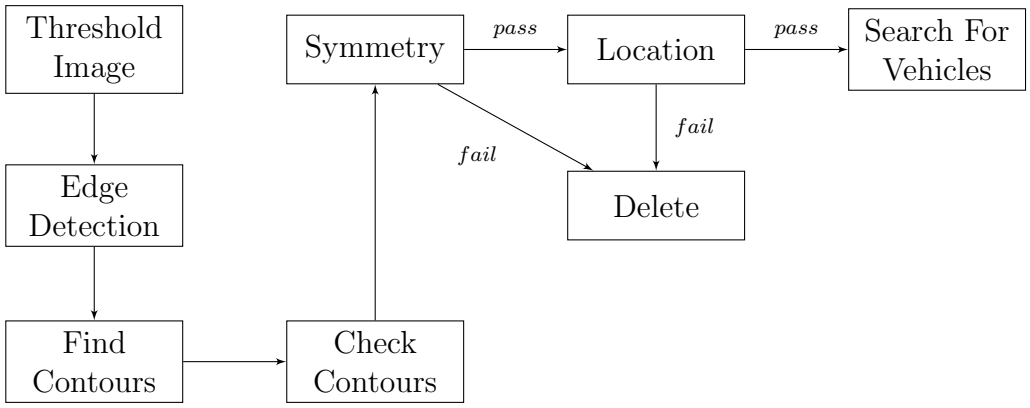

Figure 4: Contour Checks

The next process uses the same Canny Edge detection from lane detection in order to provide the findContours function from OpenCV the proper image format to scan for closed contours. After scanning for contours, another set of characteristics are required of each contour in order to move to the next round of processing. These characteristics are symmetry and location. Symmetry is an important characteristic because even from a different lane, the contour of a vehicle's shadow is much more symmetric than the shadow of a tree line, the major false positive that this characteristic check tries to eliminate. Symmetry is determined by finding the specific region in the threshold image that we are checking. A copy of the image is then made and rotated about the y axis. This rotation is necessary because the shadows should be vertically symmetric. The two threshold images are then combined such that overlapping dark and light areas will appear dark, overlapping dark

areas will appear dark, and overlapping light areas will appear light. The total number of dark pixels is calculated for this image and the original region. If the number of dark pixels are within a certain threshold then the contour is determined to be symmetric.

The second characteristic that is checked is location. Specifically, we check that the width of the contour is within a certain range of the expected value for its location in the image. This heuristic is based on the fact that as a vehicle moves further away from the camera the size of the vehicle decreases. Also, as the vehicle moves away, it appears to move closer to the top of the image. There is a relationship between these two characteristics that can be exploited to allow for a calculation to be made to give the expected width of a contour based on its distance from the top of the image. It is this expected width value that is used to determine if the contour found could possibly be a vehicle. This width to distance ratio can be used for all vehicles since their widths are similar, however, it will not work when trying to detect motorcycles as their width is outside of the allowable range.

4.2.3 Rescale Regions of Interest

The next portion of vehicle detection searches regions specified by the contours from the previous step. An example of the contour region can be seen in Figure 4 (a). The bounding box of each contour is used to expand the region around the contour in order to look for vehicles. An example of the region expansion can be seen in Figure 4 (b). This is necessary because

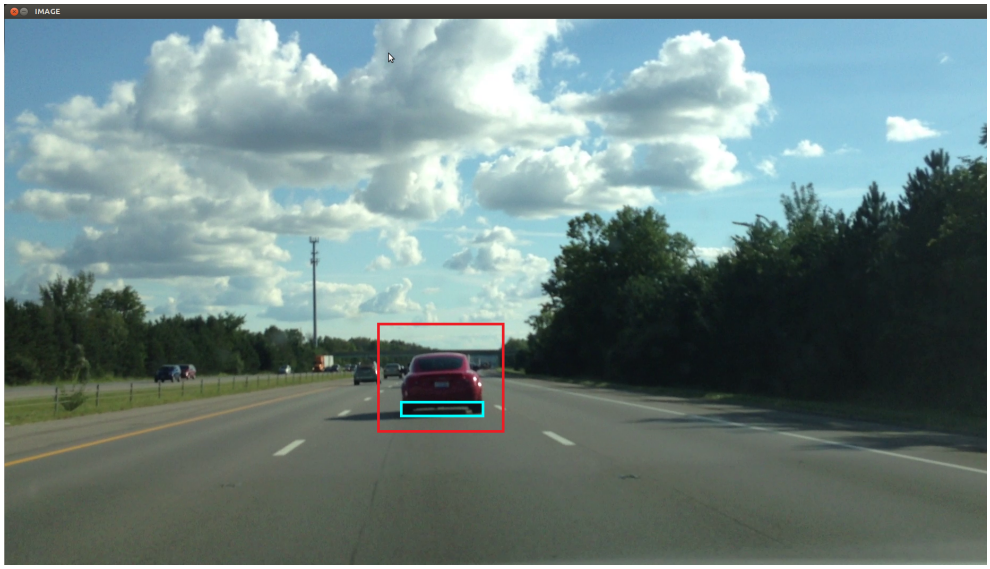

Figure 5: The cyan rectangle is the bounding box of a detected contour. The red rectangle is the expanded search region based on the dimensions of the bounding box.

the contour only captures the underside of a possible vehicle and does not provide enough information to confirm or deny that the region contains a vehicle. The region is then expanded through a predetermined ratio adjustment that is based on of the characteristic that most vehicles are relatively square when viewed from behind. This new region is then scanned to find the top of the vehicle in the next step.

4.2.4 Detect Top of Vehicles

The final step in hypothesis confirmation is determining the top of each vehicle. This is done by scanning, again, for Canny Edges and Hough Line Transforms. This provides the algorithm with a list of lines in the region.

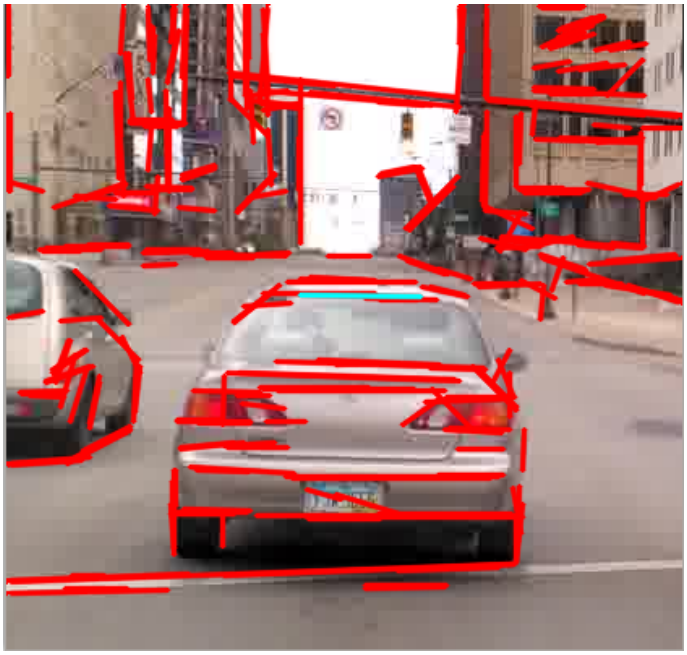

Figure 6: Image shows all lines detected with Hough Transform (red) and the best line (cyan). All lines are scaled based on position and original length.

This list is then reduced by removing lines that are not horizontal enough based on the vertical range of the segment. Each line of the remaining list is then run through a formula that scales it based on its distance from the middle of the image. The middle of the image was chosen for this because it is where the tops of most vehicles appear. The longest of the scaled lines is then returned if it meets a minimum length requirement. This minimum requirement is necessary to remove any last regions that do not contain a vehicle. These regions are dropped because most remaining false positives are just dark areas from the image and do not have many lines, let alone lines that are long and horizontal. An example in Figure 5 shows all detected lines

in red and the selected best line in cyan.

4.3 Vehicle Tracking

4.3.1 Merge Frames

The first step in vehicle tracking is merging information from the two frames. This process involves removing any regions that overlap across frames and updating their scores accordingly. To do this, each region from the previous frame is compared to each region in the current frame. If two regions overlap by a significant amount then their sizes are compared. If the dimensions of the regions are similar enough then the current frame region is kept and the previous frame is deleted. Also, the score of the kept region is increased to reflect the increase in confidence that it contains a vehicle. If the dimensions are not similar enough then the current frame region is deleted and the previous frame region is kept. The region's score is then decreased to reflect the decrease in confidence due to a missed detection in the current frame. In the next step of vehicle tracking, any region that was not matched from the previous frame must be found in the current frame. To do this, the region is expanded and a feature detector is used to detect the vehicle in the current frame. However, the feature detector needs an image to compare to. The image used in this step is the most recent region found during regular vehicle detection and is called the check image. To prevent introducing errors in the next step, the check image is only updated when a region is detected in the current frame and not when it is tracked. Therefore, the check image for each region still on the list from the current frame is updated to be its

current region.

4.3.2 Correcting Missed Regions

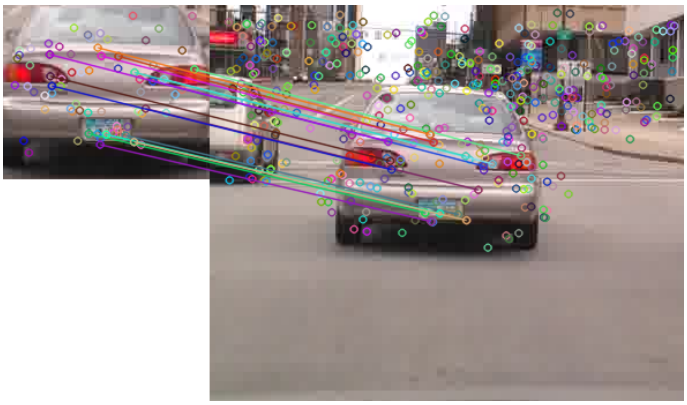

Figure 7: Left image is the check image, right image is the current frame image. Keypoints in both images are designated by circles. If a match between keypoints in different images is found, then a line is drawn between the points and the pair is stored.

At this point, the list of regions from the previous frame are any regions that were not detected in the current frame. It is assumed that these regions still contain vehicles but the position information of each frame is old and needs to be updated. To do this, each region is expanded and feature detection is used on this new region to detect the new position of the vehicle. Our algorithm uses a SURF based feature detector and descriptor extractor. SURF is a robust and fast feature detector preferred for object recognition. A brute force matcher is then used to match the keypoints of the check image to the expanded image. This technique is useful because feature detectors select edges and corners as keypoints which are both a prominent feature of

vehicles. However, there is a need to remove outliers from the result due to the fact that all keypoints must be matched in a brute force approach. This pruning is done by finding the minimum distance of all pairs. Any pairs that are farther apart than two times the minimum distance are removed. Each matching pair found is then compared to find the translation of the vehicle in the current frame. An example of the matching of keypoints can be seen in Figure 6.

To calculate how much each vehicle moves, the average change in the x and y coordinate is found for each region. If the translation is considered too high, then the algorithm assumes that there was a bad translation, uses the old region as the new region, and decreases the score for that region. If the translation is a reasonable value then the region is rescaled based on the original region and the translation amount. Next, if the resulting region has an expected width to height ratio the translation is considered good, the new region is used, and the score is increased. If the width to height ratio is not within reason, then the translation is considered bad, the old region is used, and the score is decreased. If no matches were found then of course the region is considered matchless, the old region is used, and the score is decreased.

After checking all regions from the previous frame, all regions are added to the current frame list. All region scores are checked and any that are negative are removed. This list is the final list for the frame.

4.3.3 Visualize

The remaining regions are then considered to contain vehicles. These regions are then printed to a copy of the current frame along with the lane markers using a Visualizer class.

5 Evaluation

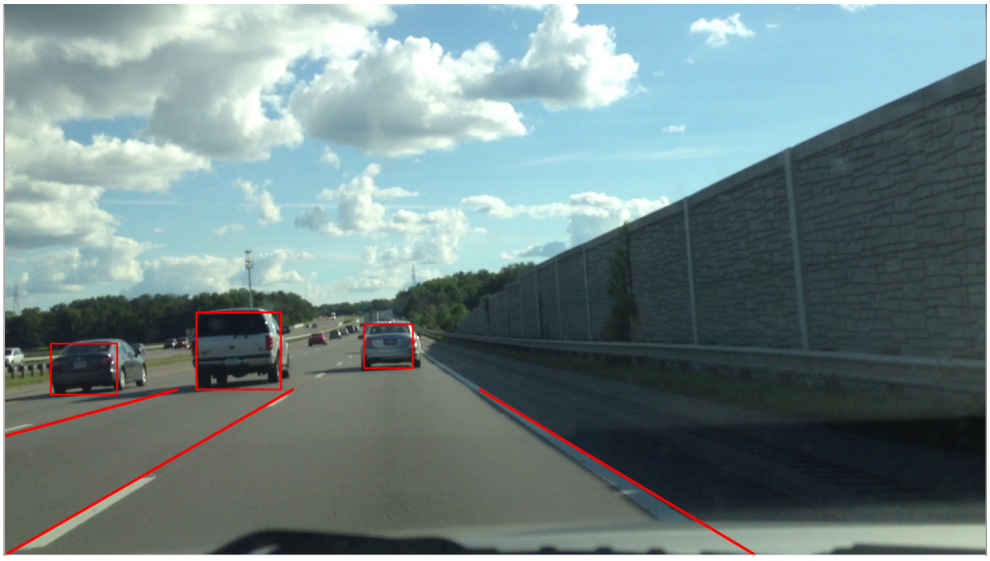

Figure 8: Successful Vehicle Detection

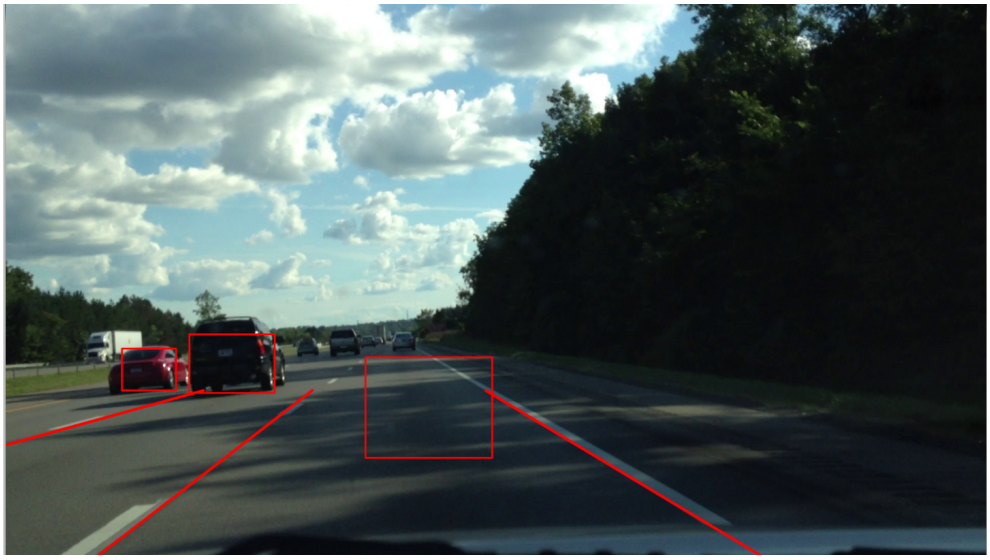

Figure 9: Unsuccessful Vehicle Detection

5.1 Test Results

Proper output of the algorithm should look like the image in Figure 7. This image locates all three vehicles within range as well as does not mark any false positives. The image in Figure 8 however, is an example of poor performance from the algorithm. Figure 8 should not have the false positive in the current lane marked and should have marked the SUV in the lane to the front-left as well as the car in the current lane.

At this point in time, testing is limited to pre-recorded videos of dashcam footage. However, this method allows for various metrics to be calculated for each video. There are two statistical metrics that express important information for this project. One is the precision:

$$precision = \frac{\text{\# of cars detected}}{\text{\# of cars detected + \# of regions that contain a nonvehicle}}$$

The second metric is recall:

$$recall = \frac{\text{\# of cars detected}}{\text{\# of cars detected + \# of cars that were missed}}$$

Because this method of testing requires user input to determine which regions are good and which are not, 100 randomly chosen frames were used to find the average precision and recall of each test. This provided a statistically significant sample from the video population while still providing an efficient method of testing. Because the algorithm cannot be expected to detect any

vehicle in the frame, regardless of distance, only vehicles that are within roughly 60 feet of the user's vehicle are considered. Based on analysis, the algorithms have achieved a precision of 82% and a recall of 76%. There is a way to improve on both of these numbers by altering the decision algorithm. A description of how to improve the algorithm is given in the next section

5.2 Future Work

There is still need to improve both precision and accuracy of the detection algorithm. The weak points of the algorithm were determined to be when using the threshold filter. Only vehicles within 30 feet of the user were detected under varying lighting conditions and vehicles between 30 and 40 feet were only detected under advantageous lighting conditions. The threshold filter can be altered to pick up vehicles that are further away by changing the threshold value. However, doing so destroys any information associated with close-by vehicles. Scanning multiple times at different threshold values would solve this problem, however, it is too slow to run anywhere near real time. Therefore, to increase the detecting range, a new technique must be implemented.

The technique proposed leverages lane detection further. Each lane will detect vehicles at its own rate instead of a frame-wide implementation. The algorithm for the current lane of the user is the simplest to implement so we will start there.

First, Canny Edge Detection is used to find all of the edges in the image.

Similar to the current image processing, the vanishing point is used to mark the top of the search region. Next, a search box is created immediately in front of the user's vehicle. This box has the dimensions of starting at the left marker for the current lane and ending at the right lane marker. The height of the box is controlled by a user variable. The larger the value, the faster the scan but the less accurate the vehicle detection will be. The box is then moved away from the user's vehicle, remaining in the center of the lane and bounded by the lane markers. Because the output image of Canny Edge Detection makes all non-edge pixels black and all edge pixels white, finding the box with the number of white pixels above a certain threshold indicates the bottom of a vehicle. This box is then returned and the current algorithm uses it like it does the bounding box for contours. One difference in the later algorithm is that there is no scan every frame so the vehicle must be located simply by tracking with feature detection.

The difference between the current lane scan and side lane scans is that there may be multiple vehicles in side lanes that must be detected. To do this, once the bottom of a vehicle is detected, the search box must move up the lane to a point where it will not detect the same vehicle again and continue scanning. Also, because vehicles from the current lane can move into side lanes there must be periodic scans to pick them up. However, the entire lane does not need to be searched as lane-changing vehicles must come from the current lane.

This proposed algorithm should be able to extend the search range at

least to the 60 foot mark used in testing if not past that.

Considering that this work was done assuming daylight conditions, there is also a need to implement vehicle detection at night. To do this would require using different vehicle characteristics. The most apparent solution would be to locate the rear lights of each vehicle. Other light sources may get detected in which case using the position and size of each light source as well as matching both rear lights of each vehicle will reduce false positives and aid in detecting the vehicles.

References

[1] United States Census Bureau. (2012). Motor Vehicle Accidents Number and Deaths: 1990 to 2009. Retrieved from https://www.census.gov/compendia/statab/2012/tables/12s1103.pdf

[2] United States Census Bureau, (2012). Distracted Drivers Crashes/Road Fatalities and Injuries: 2005 to 2009. Retrieved from https://www.census.gov/compendia/statab/2012/tables/12s1109.pdf

[3] Howard, B. (2013). Blind Spot Detection: Car tech that watches where you cant. Retrieved from http://www.extremetech.com/extreme/165742-blind-spot-detection-car-tech-that-watches-where-you-cant

[4] Guizzo, E. (2011, October 18). How Google's Self-Driving Car Works. Retreived from http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works

[5] OpenCV, http://opencv.org/

[6] Betke, Margrit, Esin Haritaoglu, and Larry S. Davis. "Real-time multiple vehicle detection and tracking from a moving vehicle." *Machine Vision and Applications* 12.2 (2000): 69-83

[7] Matthews, N. D., et al. "Vehicle detection and recognition in greyscale imagery." Control Engineering Practice 4.4 (1996): 473-479

[8] Parodi, P., and G. Piccioli. "A feature-based recognition scheme for traffic scenes." Intelligent Vehicles' 95 Symposium., Proceedings of the. IEEE, 1995

[9] Wu, Junwen, and Xuegong Zhang. "A PCA classifier and its application in vehicle detection." Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on. Vol. 1. IEEE, 2001.

[10] Tuohy, S., et al. "Distance determination for an automobile environment using inverse perspective mapping in OpenCV." Signals and Systems Conference (ISSC 2010), IET Irish. IET, 2010.

[11] Sun, Zehang, George Bebis, and Ronald Miller. "On-road vehicle detection: A review." Pattern Analysis and Machine Intelligence, IEEE Transactions on 28.5 (2006): 694-711.

[12] Hancock, J. "High-speed obstacle detection for automated highway applications" Tech. Rep. RI-TR-97-17, Carnegie Mellon University, 1997.

[13] Franke, U., Kutzbach, I. "Fast stereo based object detection for stop&go traffic" *Intelligent Vehicle*, pp. 339-344, 1996.

[14] http://www.ionroad.com/

[15] http://www.mobileye.com/